Terms used **IMS database**

Found **5** of **3,676** **searched out of 3,676.**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search

Try this search in The Digital Library

Results 1 - 5 of 5

Relevance scale ■■■■■

1 Physical design equivalencies in database conversion

Mark L. Gillenson

August 1990 **Communications of the ACM**, Volume 33 Issue 8

Full text available: pdf(1.45 MB) Additional Information: full citation, abstract, references, citings, index terms, review

As relational technology becomes increasingly accepted in commercial data processing, conversion of some of the huge number of existing navigational databases to relational databases is inevitable. It is thus important to understand how to recognize physical design modifications and enhancements in the navigational databases and how to convert them to equivalent relational terms as applicable.

Keywords: database performance

2 Applying AI to Software Renovation

Robert E. Filman

July 1997 **Automated Software Engineering**, Volume 4 Issue 3

Full text available: Publisher Site Additional Information: full citation, abstract

Lockheed Martin InVision provides software renovation and sustainment services, including analyzing systems for "interesting features," transforming systems to new environments, and recasting systems to new architectures and languages. We seek an optimal blend of effort by automating the straightforward parts of a reengineering task under human control. We achieve this automation through a judicious combination of artificial intelligence and compiler-compiler techniq ...

Keywords: modernizing legacy systems, software maintenance, software reengineering, software renovation, software sustainment

3 Empirical results on locality in database referencing

A. Inkeri Verkamo

August 1985 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1985 ACM SIGMETRICS conference on Measurement and modeling of computer systems**, Volume 13 Issue 2

Full text available: pdf(933.05 KB) Additional Information: full citation, abstract, references, citings, index terms

Database referencing behaviour is analyzed with respect to locality features. The analysis is based on database reference strings collected from several runs of typical batch programs accessing a real database. Locality of reference is measured by the stack distance probability

distribution, the number of block faults, and a locality measure based on the memory reservation size. In all the experiments, locality of reference is observed, but it is found to be weaker than in code r ...

4 Database design: A practical design methodology for the implementation of IMS databases, using the entity-relationship model

Ewing L. Lusk, Ross A. Overbeek, Bruce Parrello

May 1980 **Proceedings of the 1980 ACM SIGMOD international conference on Management of data**

Full text available: pdf(967.85 KB) Additional Information: full citation, references, citings

5 Disk cache—miss ratio analysis and design considerations

Alan J. Smith

August 1985 **ACM Transactions on Computer Systems (TOCS)**, Volume 3 Issue 3

Full text available: pdf(3.13 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The current trend of computer system technology is toward CPUs with rapidly increasing processing power and toward disk drives of rapidly increasing density, but with disk performance increasing very slowly if at all. The implication of these trends is that at some point the processing power of computer systems will be limited by the throughput of the input/output (I/O) system. A solution to this problem, which is described and evaluated in this paper, is disk cache

Results 1 - 5 of 5

Your search matched **6** of **1043368** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance** in **Descending** order.

Refine This Search:
You may refine your search by editing the current search expression or entering a new one in the text box.

'database reorganization' Search

☐ Check to search within this result set

Results Key:
JNL = Journal or Magazine **CNF** = Conference **STD** = Standard

---

1 **Database reorganization in parallel disk arrays with I/O service stealing**
Zabback, P.; Onyuksei, I.; Scheuermann, P.; Welkum, G.;
Knowledge and Data Engineering, IEEE Transactions on , Volume: 10 , Issue: 5 , Sept.-Oct. 1998
Pages:855 - 858

[Abstract] [PDF Full-Text (60 KB)] IEEE JNL

---

2 **ADDROL: a method for on-line reorganization in a multisystem DBMS with shared disks**
Wietrzyk, V.I.S.; Orgun, M.A.;
Parallel and Distributed Systems: Workshops, Seventh International Conference on, 2000 , 4-7 July 2000
Pages:378 - 383

[Abstract] [PDF Full-Text (408 KB)] IEEE CNF

---

3 **Transactional aid to workflow distribution to support e-commerce**
Wietrzyk, V.I.; Takizawa, M.; Enokido, T.; Grosky, B.;
Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on , 1-5 Sept. 2003
Pages:199 - 203

[Abstract] [PDF Full-Text (306 KB)] IEEE CNF

---

4 **Applicability of IEEE maintenance process for corrective maintenance outsourcing-an empirical study**
Rao, B.S.; Sarda, N.L.;
Software Maintenance, 2002. Proceedings. International Conference on , 3-6 Oct. 2002
Pages:74 - 83

[Abstract] [PDF Full-Text (383 KB)] IEEE CNF

---

5 **Optimal database reorganization policies: a stochastic control approach**
Park, J.S.; Bartoszynski, R.; Pirkul, H.;
System Sciences, 1989. Vol.III: Decision Support and Knowledge Based Systems Track, Proceedings of the Twenty-Second Annual Hawaii International Conference on , Volume: 3 , 3-6 Jan. 1989
Pages:752 - 761 vol.3

[Abstract] [PDF Full-Text (720 KB)] **IEEE CNF**

---

6 **A high performance configurable storage manager**
Biliris, A.; Panagos, E.;
Data Engineering, 1995. Proceedings of the Eleventh International Conference on , 6-10 March 1995
Pages:35 - 43

[Abstract] [PDF Full-Text (892 KB)] **IEEE CNF**

Home | Log-out | Journals | Conference Proceedings | Standards | Search by Author | Basic Search | Advanced Search | Join IEEE | Web Account | New this week | OPAC Linking Information | Your Feedback | Technical Support | Email Alerting | No Robots Please | Release Notes | IEEE Online Publications | Help | FAQ| Terms | Back to Top

US-PAT-NO: 6519612

DOCUMENT-IDENTIFIER: US 6519612 B1

TITLE: Internet storage manipulation and navigation system

DATE-ISSUED: February 11, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Howard; David E.	Loveland	CO	N/A	N/A
Gandee; John J.	Windsor	CO	N/A	N/A
Godwin; Kurt E.	Loveland	CO	N/A	N/A

US-CL-CURRENT: 707/200, 707/102 , 707/3

ABSTRACT:

The parallel virtual directory system can extend the native file system to provide a superior method for organizing a computer system's physical storage devices or locations. These can include hard disks and removable media or remote storage locations such as on the Internet. Interceptor modules can monitor all file or memory changes such as creations, writes and deletes to the native file system and can pass this information to the parallel virtual directory system. The parallel virtual directory system may be accessed through the native file system methods allowing users to view their file system as it existed at any point in time, including removable media that is no longer present in the System Further, via the parallel virtual directory system and the Interceptor modules, users can access any file in any of the monitored file systems. The parallel virtual directory system may be implemented using database technology thereby allowing multiple views of the parallel virtual directory system to be offered to the user based on their needs and can thus be used to ease navigation through the file system. Further, a set of information management processes can run at the application level providing data management services such as backup, archiving, recording of digital video programming, or even controlled playback or utilization of that information. The information management processes can configure the parallel virtual directory system to capture the information they require and use the gathered information to efficiently complete their data management functions.

124 Claims, 28 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 24

---------- KWIC ---------

Brief Summary Text - BSTX (4):

The hierarchical directory systems are intended to allow users to organize their data in ways that makes it easier for them to understand. This is done by creating directories, or folders, for specific classes of data such as word processing documents or computer programs. Once a directory structure for the storage system is defined it becomes a tedious and difficult task to **reorganize** the directory structure. Also, files of the same name cannot be stored within

the same directory due to the common need for files to be differentiated by unique names even though it is common for users to desire to save copies of the same file at different points in the evolution of the file. For example, it would be very convenient if a writer could keep the same file name for an article and differentiate the various versions of the article by the date and time of creation of each version rather than having to use minor variations on the original name.

Brief Summary Text - BSTX (17):

An important benefit of the invention is that the virtual directory system also can connect to the native file system of the computer system in such a way that the virtual directory system appears to be a physical storage device. However, it has the advantage that if a user elected to the virtual directory system could represent all of the files contained on all of the physical file storage devices connected to the computer system. Because the files stored on the physical storage devices can be represented in the virtual directory system and because the virtual directory system has among its characteristics the qualities of a relational database, the file information for the various files can be **reorganized,** reconfigured, or searched easily. These features allow the user greater flexibility in how files are viewed and how the user finds data. Additionally, the database can overcome the requirement of unique file names, as imposed by traditional hierarchical directory systems. Multiple instances of a given file name can now be presented in the same directory with the uniqueness of the file data determined by the time the file was modified, the media it was stored on, or some other file characteristic.

Drawing Description Text - DRTX (20):

FIG. 19--Shows a directory structure for a virtual directory database configured to present three **separate** virtual directories.

Detailed Description Text - DETX (5):

The virtual directory system can serve many functions and appear in different embodiments. Perhaps one of its most useful functions is the ability to store file attribute information (16) about files stored on a native file directory system (8) and present hat file information as if the files were stored in a distinct directory **separate** from the native file directory system.

Detailed Description Text - DETX (8):

Furthermore, because the virtual directory system can be stored on a configurable database, the directory itself can be subdivided into additional virtual directories. Hence, a user can group all of the word processing applications into a single directory, all of the letters to relatives (which also are stored in the word processing directory) in a second directory, and all of the letters to Aunt Margaret (which are also stored in each of the first two directories) in a third directory. Then, each of these directories can be utilized by application programs as if they were directories of actual physical file storage devices, such as a local hard drive. The application programs which utilize the file information do not need to be able to configure the database of the virtual directory, rather the directory can be configured at the operating system level and therefore it can appear as a physical storage device when called by the application level programs that need files represented by the virtual directory system. (It should be understood, however, that the virtual file directory system can be configured by an application level program. Therefore, the configuration can be accomplished quite easily when a new configuration is desired.) While prior application programs may have been able to search a native file directory system in order to create a desired grouping of files, these application programs did not make the groupings available for other application programs. For example, search of a directory system by a WordPerfect.TM. word processing program might have created a listing of files of all of a company's annual reports; however, that

listing was not then accessible by a WORD.TM. word processing program as a **separate** directory. Rather, the WORD.TM. program would have to repeat the search, as would the WordPerfect.TM. program when it was reimplemented. Hence, the ability to configure the virtual file directory at the operating system level is a significant feature of this embodiment of the invention. Others have apparently failed to appreciate the ability to accomplish this important feature or may have believed that it was not functionally implementable do to hardware constraints. However, with the development of new operating systems, the embodiments of the present invention will and are implementable to accomplish greatly improved performance over present rigidly structured directory systems.

Detailed Description Text - DETX (25):

FIG. 11 shows a flowchart of one possible method of updating the virtual directory with new file information. A native file directory can be presented (300). Such a native file directory will typically be hierarchical in nature. A "hierarchical directory is intended to mean a directory with a rigid structure of file attribute information, the structure of which cannot be reconfigured after its selection. For example, a hierarchical directory could be comprised of a number of segment types arranged in a hierarchy, commencing with the **root segment** type; below the **root segment** type there is zero, one, or more segment types at the first level, with a similar structure below each of these first level types at the second level, and so on. Thus each segment type is dependent on a segment type at the immediately higher level. As an additional ample, the typical structure of one's C: hard drive is envisioned as displaying drive name/directory/subdirectory/ . . . /file name. Whereas a non-rigid hierarchy would allow presentation of this information in other manners, such as file name/subdirectory/. . . /directory/drive name. In addition, a virtual directory file system can be presented (304). Then, the I/O system of the computer system can be monitored by an Interceptor (2) for input/output commands or requests to the native file system (308). If the Interceptor detects such a request, the virtual directory system can check whether it wants to track the change--which might be determined based on the target storage device of the request. If the request is to be tracked, the request can be checked to see if it is a file save request (316). If it is, then the file attributes being passed through the interceptor to the native file system storage device can be captured (or copied) (320) and passed to the virtual file directory in order to update the virtual file directory with the file attribute information (324), e.g., by storage as a new record in the database. If the request is a fie read request, the interceptor could capture the name of the application program that originates the file read request and then use that information to update the virtual directory system. Several of these functions might occur from the virtual directory system itself or might also be accomplished by the Interceptor independently. FIG. 13, for example, indicates that the monitor element which monitors for changes made to the native directory system (68) could be comprised of a code module that is part of the Interceptor. Indeed, for some embodiments, the Interceptor could be considered a component of the virtual directory system. FIG. 7 shows a Windows.TM. 95 implementation of hooking file information and updating a database. FIG. 9 and FIG. 12 show alternative implementations of the file and media monitoring functions, for example, for a Windows 95 implementation.

Detailed Description Text - DETX (29):

As noted earlier, some of the capabilities of this paarallel virtual directory system which can provide parallel information of a native directory system aspect of the invention may include providing the ability to connect to the native operating system's file system interface in a manner similar to a block device, such as a hard disk. It may allow read/write access to all victual files. In addition the system may utilize database technology to allow gross manipulation of the information in a gross manner to permit **reorganization,** recharacterization and other appropriate handling or presentation of the information. It may also present to the native operating

system a directory structure that is composed of entries gathered via the Interceptor module(s), and may present to the system user a virtual file system representative of all files contained within the native file systems. The virtual directory system may provide a secondary interface so that application level programs may configure its database so that multiple views of the global system may be created. Through an ability to present multiple views, it may export those views to the operating system's file system interface in such a way that the views are presented to the end user as discrete logical devices via the native file browsing mechanisms. In utilizing its database capabilities, the global system may organize lists of files that meet predefined, or user defined criteria and may provide these lists of files on demand to Information Management Processes (IMPs) to more efficiently carry out data management processes. It may also allow the end user to access any file in the computer system, either directly through the native file system, or indirectly through the parallel virtual directory system.

Detailed Description Text - DETX (34):

In one embodiment of the invention, the interceptor can be inserted between the operating system interface to the physical storage devices and the physical storage devices themselves. This might be visualized as a single interceptor (2) module as shown in FIG. 1 or as **separate** interceptor modules (2) for each storage device as shown in FIG. 2. The interceptor module can be programming code that receives an Input/Output command from the I/O interface and determines from that command whether the interceptor (or its master, such as the virtual directory system) is interested in the information being input to or output from the native file directory. The Interceptor (2) might even be used to initiate its own command to check system flags, interrupts, or system status, when the I/O command is received. In this manner, it could be used as an intelligence gathering or error checking device for a system. If the interceptor is interested in a command, it can then relay the information to its master, such as a second file directory system (88) like the virtual directory system. Such information might include the name of the application program that sent a command or request, or file attribute information (16) for a file. This information could then be stored in the second file system. As shown in FIG. 13, such processes could be accomplished by programming code that acts as: a monitor element which monitors input/output (108), a relay element which relays intercepted information (96), a captures element which captures application program name (124).

Detailed Description Text - DETX (35):

Additionally, or as a **separate** embodiment of the invention, the Interceptor (2) could be used to monitor for the attachment of storage media that was either reconnected to the computer system or which was newly connected to the computer system. If there were such a desire, the Interceptor could even be used to monitor when media was removed from the computer system. This process can be understood by reference to FIG. 8 which was discussed earlier. For example, it can be used to check the new media for any new information that should be added to the virtual directory system. This might be accomplished by accessing the directory structure of the new media, scanning for any new information (i.e. traversing and comparing to the data in the second file system) and relaying the data. Program code modules as shown in FIG. 13 could be used to accomplish the scanner element which scans (104), monitor element which monitors I/O requests (108), copier element which copies file path information (112), relay element which relays information (96), and checker element which checks time stamp information (120).

Detailed Description Text - DETX (46):

For example a typical DOS directory is shown in FIG. 20. This directory is shown as an extended version in FIG. 21. An extended directory of information in the virtual directory file system designated as H: is shown in FIG. 21. This extended system has added a description field and three backup fields

where files can be backed up when they are changed. Furthermore, the extended directory demonstrates that files can exist with the same file name in the same directory. The database can distinguish such files based on other file attributes such as time and date of creation or file size if it were configured to do so. Because the virtual database is **separate** from the native file directory, it does not destroy the native file directory system. Rather, the native directory system can be accessed as always; while the virtual directory file database can permit more advanced file access.